In [ ]:

```
# Chris Straschewski
# CMS180014
# CS 4395.001

import sklearn
import csv
import pandas as pd
from sklearn.naive_bayes import MultinomialNB

# Dataset used
# SMS Spam Collection (Text Classification)
# https://www.kaggle.com/datasets/thedevastator/sms-spam-collection-a-more

## Naive Bayes First

# load dataset
df = pd.read_csv('IMDB Dataset.csv', header=0, usecols=[0,1], encoding='la

df[df == 'positive'] = 1
df[df == 'negative'] = 0

print('rows and columns:', df.shape)
print(df.head())

# Text Preprocessing
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=list(stopwords))



# set up X and y
X = df.review
Y = df.sentiment
Y = Y.astype('int')

# take a peek at X
print(X.head())

print(Y[:10])

# split data
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, t

print(X_train.shape)

# apply tfidf vectorizer
X_train = vectorizer.fit_transform(X_train)  # fit and transform the train
X_test = vectorizer.transform(X_test)        # transform only the test dat

# take a peek at the data
print('train size:', X_train.shape)
print(X_train.toarray()[:5])
print('\ntest size:', X_test.shape)
print(X_test.toarray()[:5])

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, Y_train)

# priors
import math
prior_p = sum(Y_train == 1)/len(Y_train)
print('\nprior spam:', prior_p, 'log of prior:', math.log(prior_p))

# the model prior matches the prior calculated above
naive_bayes.class_log_prior_[1]

# what else did it learn from the data?
# the log likelihood of words given the class
print('\n', naive_bayes.feature_log_prob_)

from sklearn.metrics import accuracy_score, precision_score, recall_score,

# make predictions on the test data
pred = naive_bayes.predict(X_test)

# print confusion matrix
print('\n', confusion_matrix(Y_test, pred))

print('accuracy score: ', accuracy_score(Y_test, pred))

print('\nprecision score (not spam): ', precision_score(Y_test, pred, pos_
print('precision score (spam): ', precision_score(Y_test, pred))

print('\nrecall score: (not spam)', recall_score(Y_test, pred, pos_label=0
print('recall score: (spam)', recall_score(Y_test, pred))

print('\nf1 score: ', f1_score(Y_test, pred))

from sklearn.metrics import classification_report
print(classification_report(Y_test, pred))

print('spam size in test data:',Y_test[Y_test==0].shape[0])
print('test size: ', len(Y_test))
baseline = Y_test[Y_test==0].shape[0] / Y_test.shape[0]
print(baseline)

# Naive Bayes Done, now we try Logistic Regression

## Logistic Regression
from sklearn.linear_model import LogisticRegression

# train
LRClassifier = LogisticRegression(solver='lbfgs', class_weight='balanced')
LRClassifier.fit(X_train, Y_train)

# evaluate
pred2 = LRClassifier.predict(X_test)
print('\naccuracy score: ', accuracy_score(Y_test, pred2))
print('precision score: ', precision_score(Y_test, pred2))
print('recall score: ', recall_score(Y_test, pred2))
print('f1 score: ', f1_score(Y_test, pred2))
probs = LRClassifier.predict_proba(X_test)
print('log loss: ', log_loss(Y_test, probs))

## Neural Networks
from sklearn.neural_network import MLPClassifier

NNClassifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
NNClassifier.fit(X_train, Y_train)

pred3 = NNClassifier.predict(X_test)
print('\naccuracy score: ', accuracy_score(Y_test, pred3))
print('precision score: ', precision_score(Y_test, pred3))
print('recall score: ', recall_score(Y_test, pred3))
print('f1 score: ', f1_score(Y_test, pred3))
```